Orienteering Algorithms for Generating Travel Itineraries

Zachary Friggstad
University of Alberta
zacharyf@ualberta.ca

Sreenivas Gollapudi
Google Research
sgollapu@google.com

Kostas Kollias
Google Research
kostaskollias@google.com

Tamas Sarlos
Google Research
stamas@google.com

Chaitanya Swamy
University of Waterloo
cswamy@uwaterloo.ca

Andrew Tomkins
Google Research
tomkins@google.com

Abstract

We study the problem of automatically and efficiently generating itineraries for users who are on vacation. We focus on the common case, wherein the trip duration is more than a single day. Previous efficient algorithms based on greedy heuristics suffer from two problems. First, the itineraries are often unbalanced, with excellent days visiting top attractions followed by days of exclusively lower-quality alternatives. Second, the trips often re-visit neighborhoods repeatedly in order to cover increasingly low-tier points of interest. Our primary technical contribution is an algorithm that addresses both these problems by maximizing the quality of the worst day. We give theoretical results showing that this algorithm's competitive factor is within a factor two of the guarantee of the best available algorithm for a single day, across many variations of the problem. We also give detailed empirical evaluations using two distinct datasets: (a) anonymized Google historical visit data and (b) Foursquare public check-in data. We show first that the overall utility of our itineraries is almost identical to that of algorithms specifically designed to maximize total utility, while the utility of the worst day of our itineraries is roughly twice that obtained from other approaches. We then turn to evaluation based on human raters who score our itineraries only slightly below the itineraries created by human travel experts with deep knowledge of the area.

ACM Reference Format:
Zachary Friggstad, Sreenivas Gollapudi, Kostas Kollias, Tamas Sarlos, Chaitanya Swamy, and Andrew Tomkins. 2018. Orienteering Algorithms for Generating Travel Itineraries. In *WSDM 2018: WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining , February 5–9, 2018, Marina Del Rey, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3159652.3159697

1 Introduction

Google aims to produce high-quality collections of information responsive to user needs. In the context of travel planning, it is often important to organize information about specific points of interest at a destination into a useful form such as an itinerary for a visit. We have developed and launched such a capability, which is used at various places in Google's product offerings. In one setting, the itinerary is pre-computed for hundreds of cities, which is then shown to users searching for travel in a particular city as an aid to jump-start the travel planning process. In another setting, users who have "saved" points of interest will see itineraries generated on the fly to feature the specific places they have saved. And in yet a third setting, users who have downloaded the Google Trips app will see dynamic editable itineraries which may be edited and reconfigured in real time to suit the user's needs. In this last setting, the algorithm must be sufficiently responsive to compute new itineraries in real time even running locally on a mobile device. Hence, the algorithmic generation of high-quality itineraries under various constraints is an important problem for Google, and one that we continue to explore.

The problem of generating tourist itineraries has been studied before, often in the guise of a combinatorial problem called the *Orienteering problem*: given a graph with costs on the edges and benefits on the vertices, find a max-benefit tour subject to a cost budget. This formulation easily captures, for example, finding a ten-hour tour of London that visits the best spots, even if the notion of "best" is personal and complex.

However, most trips range from 3–8 days, and many are even longer. The orienteering problem must therefore be extended to cover multiple days. This seemingly innocuous extension in fact raises a host of new issues. First, the problem tends to be larger in scale, as one can visit many more places during a longer trip. Second, standard greedy approaches perform well from a worst-case algorithmic standpoint in some settings, but do not yield good user experience for multi-day trips. For example, the first day of a visit might cover all the best destinations, then each subsequent day might visit places one tier more boring than the day before. Worse yet, each of these days may revisit the same neighborhoods along the same routes, simply replacing high-quality stops with lower-quality ones. Some heuristics may be employed to re-order days, encourage good balance, and penalize re-visits, but as our experience has shown, these heuristics are difficult to tune appropriately across the wide range of cities that human civilization has produced.

A better approach is to re-formulate the problem to make all days of the tour good, by maximizing the benefit of the worst day rather than maximizing the sum of benefits of all days. This will avoid the issue of good versus bad days, and will also naturally encourage the algorithm to spend time one day focusing on a range of good and bad places within a particular neighborhood, moving on tomorrow to a fresh new area.

We present a simple algorithm and a proof showing its competitive factor is not much worse than the factor of whatever algorithm

WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA

ACM ISBN 978-1-4503-5581-0/18/02.
https://doi.org/10.1145/3159652.3159697

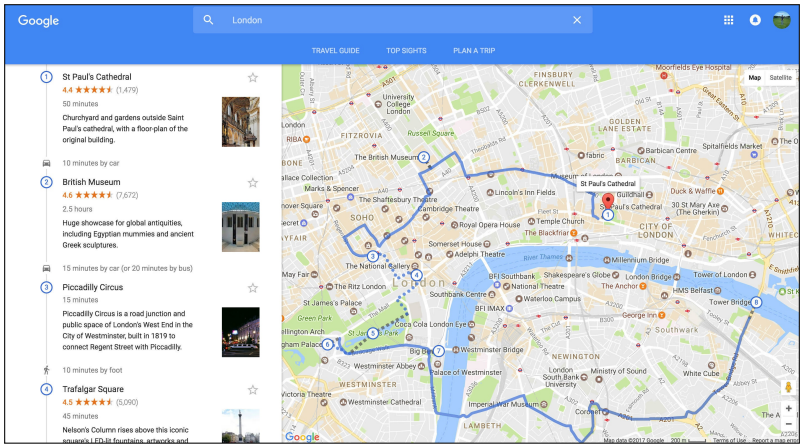

Figure 1: A precomputed tour of London.

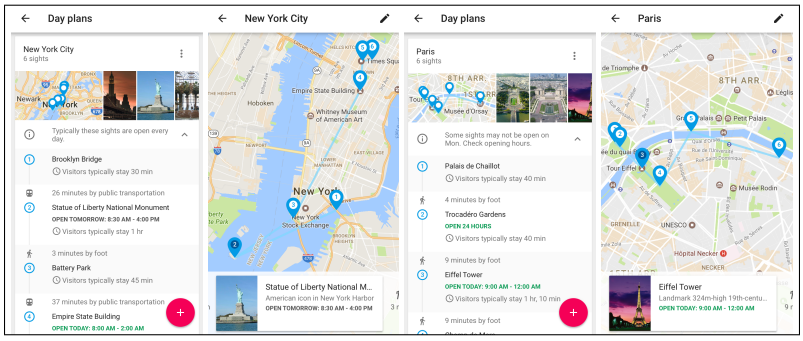

Figure 2: Tours of New York City and Paris generated on mobile.

is available to optimize a single-day visit. In particular, we show that, given an α approximation algorithm for various single-day orienteering models, our algorithm achieves an $\alpha+2$ approximation for maximizing the value of the worst day. The models covered by this result include the simplest orienteering model where the graph is undirected and there are no restrictions on obtaining the reward of a node (where we achieve a $4+\epsilon$ approximation) as well as extensions to orienteering with deadlines, or more generally, time-windows for the nodes and extensions to directed graphs (for which we achieve polylog approximations).

In response to the criticism that, while the greedy algorithm may perform well in some worst-case settings, this does not guarantee that users will enjoy the results, we provide extensive empirical verification of the performance of our algorithm and a benchmark set of alternatives. Our analysis selects two families of measures to apply to the algorithms. The first are computable measures of quality based on the objective: number of places visited, total benefit from those places, and so forth. We also add a second class of measures based on human evaluation: we employed a pool of raters tasked to assess the quality of the algorithms generated by the different algorithms. Our experimental analysis exhibits that our algorithm manages to improve the value of the worst day to a significant extent compared to a well established multi-day orienteering algorithm that maximizes the total reward obtained (on average our algorithm more than doubles the value of the worst day for 5-day itineraries across 200 cities) while at the same time sacrificing no value with respect to the total reward. In the human rater evaluation, our algorithm nearly matches the ratings of a set of multi-day itineraries curated by travel experts.

2 Related Work

Algorithmically, the orienteering problem is an excellent model for the application of planning a tourist itinerary in a city. In orienteering, the goal is to maximize the reward extracted by visiting nodes in a graph, starting and ending at a given node, subject to a budget constraint on the length of the tour. The best known approximation algorithms for orienteering and orienteering with time windows (i.e., when the reward of a node is extracted if and only if its time of visit is within a prescribed time window) are given in [7]. These algorithms follow work in [2] and [1], where the orienteering problem is solved by a dynamic program that uses the k-path algorithm from [6] as a subroutine. In the k-path problem, the goal is to minimize the cost of a path that visits at least k nodes. In turn, the work in [6] relies on a primal-dual algorithm for k-MST [15], which is the problem of constructing a tree of minimum cost that spans k nodes. The algorithm for the undirected case with no time windows is a $2+\epsilon$ approximation, while the algorithms for the other models are poly-logarithmic approximations. We discuss the best known

approximation factors for various orienteering models in more detail in Section 3. A different approach to orienteering on undirected graphs without time windows was given very recently by Friggstad and Swamy [14]. The idea is to define a linear program that encodes orienteering and use a rounding algorithm to come up with a solution that is a 3 approximation to the optimal. On multi-tour orienteering, the work in [2] presents a simple greedy algorithm that uses a single-tour algorithm as a subroutine and approximates the total reward across days almost without any loss to the approximation factor of the single-day case. An extension to submodular rewards has been studied in [8], where a quasipolynomial log factor approximation is described.

As far as the application domain is concerned, various approaches to the tourist itinerary planning problem have been proposed. The authors in [3, 16, 17, 29] present comprehensive surveys of heuristic and algorithmic approaches to the problem. The work most closely related to ours is presented in [9]. The authors there focus on generating tourist itineraries based on data extracted from social media, in particular Flickr photos. They also model the problem as an orienteering instance and use the photos dataset to extract the popularity of a place, the duration of visits, and the transit time between different points of interest. We now explain the main differences of our work to the one in [9]. First, we work with a fundamentally different dataset, specifically, the work in [9] seeks to extract the number of visits to an attraction, the visit durations and the transit times from Flickr data, whereas our dataset explicitly includes this information. Secondly, the authors in [9] focus primarily on single day itineraries and mention the multi-day case as an interesting extension, whereas our main focus is on multi-day itineraries: we propose an extension of orienteering to the multi-tour case where the objective is to optimize for the value of the worst tour and present original algorithmic work for that problem, as well as experimentally evaluate it in a large number of cities world wide.

Other interesting papers in the domain of itinerary planning include [34], where the authors use GPS data to build an itinerary recommendation engine and evaluate it using Beijing as an example, [26], which focuses on suggesting routes in a city that also offer some utility to the user as opposed to just being the shortest source-destination paths, as well as various approaches that use geo-tagged social media, e.g., [4, 21, 24, 25, 30, 32, 33, 35], and approaches based on personalization [5, 11, 12, 18–20, 22, 23, 27, 28]. A final interesting piece of related work is [31] where the authors study the orienteering problem in a tourist application from a game-theoretic view-point. They consider settings such as large amusement parks where many travelers visit the same attractions and there are congestion effects at the attractions and on the travel routes from one attraction to another. The objective of the work in [31] is to compute good equilibrium solutions to the problem.

3 Orienteering Model and Preliminaries

Consider a complete (either directed or undirected) graph $G = (V, E)$ with a designated start and end node s, visit duration costs d_v for every $v \in V$, and travel time costs c_e for every $e \in E$. We assume the edge costs satisfy the triangle inequality. For any given path P, we write $V(P)$ for the set of nodes in P and $E(P)$ for the set of edges in P. We say that a vector of node visit times τ^P is admissible if for every edge $(u, v) \in E(P)$, with $v \neq s$, it is the case that

$$\tau_v^P \geq \tau_u^P + d_u + c_{(u,v)}.$$

In *multi-tour orienteering*, our input consists of a graph G with a designated start and end node s, a number k (the number of requested tours), and a (per-tour) cost budget B. For the start/end node s, we prescribe (for simplification and without loss of generality) that the duration cost d_s and the visit time τ_s^P is always 0.

A feasible output is a set of k tours $\mathcal{P}$ and a set of corresponding admissible visit time vectors τ, such that each pair of tours intersect only at s, i.e., for all $P_1 \neq P_2 \in \mathcal{P}$,

$$V(P_1) \cap V(P_2) = \{s\},$$

and, for each tour $P \in \mathcal{P}$, the budget constraint is satisfied i.e., if $t(P)$ is the last node of P before returning to s, then,

$$\tau_{t(P)}^P + d_{t(P)} + c_{(t^P, s)} \leq B.$$

For a tour P and corresponding admissible visit time vector τ^P, the *utility function* $U\left(P, \tau^P\right)$ outputs the induced utility, which is given as follows. The utility of any node $v \in V$ is a fixed number u_v, which is obtained if and only if the visit time of the node τ_v^P is within a prescribed non empty time window w_v, i.e.,

$$U\left(P, \tau^P\right) = \sum_{v \in V(P)} u_v \cdot 1_{\{\tau_v^P \in w_v\}}.$$

In *sum* orienteering, the objective is to maximize

$$SUM(\mathcal{P}, \tau) = \sum_{P \in \mathcal{P}} U\left(P, \tau^P\right),$$

while in *max-min* orienteering the objective is to maximize

$$MIN(\mathcal{P}, \tau) = \min_{P \in \mathcal{P}} U\left(P, \tau^P\right).$$

Mapping the model to our application, the graph represents a city that a traveler visits, with the nodes being the various points of interest (POIs) of the city and the edges being the travel routes between them. The utility of a node is determined by the corresponding POI's tourist popularity and the duration cost is the expected time one should spend there to enjoy what the POI has to offer. The cost of an edge is the transit time required to move from the POI corresponding to its first endpoint to the POI corresponding to its second endpoint. The start node corresponds to the traveler's hotel (or other accommodation) where the tour needs to begin and end, and the number of paths k corresponds to the number of days spent in the city. Time 0 corresponds to the time the tour may start at each day (e.g. 9am or 10am) and time B (the budget) corresponds to the time the user needs to return to the hotel.

Single-Tour Orienteering. The sum and max-min models obviously coincide when $k = 1$, i.e, for single-tour orienteering. The multi-tour algorithms we will discuss in Section 4 make use of a single-tour oracle and for this reason, we next survey the best known single-tour algorithms.

For the case when the graph is undirected and there are no time windows, i.e., $w_v = [0, \infty)$ for all $v \in V$, the best known algorithm achieves a $2 + \epsilon$ approximation, for any $\epsilon > 0$ [7]. For undirected graphs with deadline constraints on the nodes, i.e., $w_v = [0, t_v]$ for every $v \in V$, the best known is an $O(\log OPT)$ approximation,

with OPT the value of the optimal solution [2]. For undirected graphs with time windows, [7] presents an $O(\max\{\log OPT, \log L\})$ approximation, with L the length of the largest time window.

Moving to directed graphs, [7] presents an $O\left(\log^2 OPT\right)$ approximation for the case with no time windows, an $O\left(\log^3 OPT\right)$ approximation for the case with deadlines, and for the case with general time windows an $O\left(\log^2 OPT \cdot \max\{\log OPT, \log L\}\right)$ approximation.

All these algorithms rely on large dynamic programs and, hence, are complicated and slow, with running times of $O(n^8)$ or worse, where $n = |V|$. Alternatively, a simpler linear-programming based algorithm of [14] for the undirected model without time windows achieves a 3 approximation.

We note that all these algorithms assume there are no node costs, which is in contrast to our setting. However, we observe that a simple transformation can convert any instance with node costs to an equivalent instance with zero node costs. Namely, for both undirected and directed graphs, increase the cost of each edge adjacent to a node by half of the node's cost. Then the cost of any tour remains the same as before the transformation. It is not hard to see that the triangle inequality property is preserved under this transformation.

Before wrapping up the section on single-tour orienteering, we describe an algorithm specifically tailored for use in practical settings, which means the algorithm is very fast (with run-times of 50ms or less) and performs well in real world instances. The algorithm greedily builds a tour by recalculating the traveling salesman tour of the so-far selected nodes plus each possible candidate node and picking the candidate that is the most cost-effective. We term this algorithm GreedyTSPCost and evaluate its quality in Section 5.2.

GreedyTSPCost Algorithm. We conclude our discussion of single tour orienteering with the description of our GreedyTSPCost algorithm, which is in fact the one we use as the single tour subroutine in the experimental evaluation of our multi-tour algorithms in Section 5.2. The algorithm is very efficient and useful in practice, even though it can't provide good worst case guarantees. The algorithm proves useful in scenarios where orienteering needs to be solved with very limited computational resources, for instance on mobile devices with no connectivity. The algorithm's behavior is reminiscent of greedy knapsack algorithms, always adding the node that offers us the biggest bang for the buck, i.e., yields the highest marginal increase to the utility of the tour, normalized by the marginal increase to the cost of the tour. We provide the details in Algorithm 1.

Algorithm 1: GreedyTSPCost

Input: Graph G, budget B, start/end node s
Output: Tour P, visit times τ^P

```
P* ←MakeSingleNodeTour(s)
do
    P ← P*
    best_margin← 0
    P* ← null
    for ∀v ∈ V do
        P' ← TSP(V(P) ∪ {v}) // Use heuristic, e.g. 2-OPT
        margin ← (U(P') − U(P))/(Cost(P') − Cost(P))
        if margin > best_margin and Cost(P') < B then
            best_margin ← margin
            P* ← P'
while P* ≠ null
τ_s^P ← 0
for e = (u, v) ∈ E(P) do
    τ_v^P ← τ_u^P + d_u + c_e
return (P, τ^P)
```

Theorem 3.1. *Algorithm 1 is an $\Omega(OPT)$ approximation, with OPT the optimal utility.*

Proof. Consider a simple instance such that all nodes have unit utilities and zero duration costs. In such instances, Algorithm 1 will pick, in each iteration, the node that has the smallest distance from the set of already selected nodes. Suppose the start/end has two outgoing edges of unit cost. One leads to a single node and the other leads to a clique of n nodes with 0 distances. The rest of the edge costs are the shortest tour distances on the graph we described so far. Let the budget be $B = 2$. Algorithm 1 might spend this budget to move to the isolated single node as opposed to the zero cost clique and extract utility 2 as opposed to $n + 1$. □

Such adversarial instances are very unlikely in real world cities. In fact, the specific structure of real world instances (which include very skewed utility distributions and where the visit times at nodes dominate the transit times through the city) is such that a greedy knapsack-style algorithm is expected to perform well. We point to the experimental evaluation of Section 5.2 and the positive results therein as further support of this intuition. With respect to the algorithm's running time, we observe that, assuming the size of the induced itinerary is always small in comparison to the size of the graph, GreedyTSPCost is linear in the number of nodes.

4 Multi-Tour Orienteering Algorithms

In this section we will present algorithms for solving sum and max-min orienteering. The algorithm for sum orienteering was described in [2]. We present it in Section 4.1 for completeness and then we proceed with our algorithm for max-min orienteering, which we present and analyze in Section 4.2. Both algorithms rely on access to an algorithm for the special case with $k = 1$, i.e., single-tour orienteering, which they use as a subroutine. We discussed the best known approximation factors for several models of single-tour orienteering in Section 3.

4.1 Sum Orienteering

A simple greedy algorithm (see Algorithm 2) solves the sum version of the multi-tour orienteering problem almost without any loss to the approximation factor from the single-tour case. More specifically, given an α approximation for the $k = 1$ case, the work in [2] shows that Algorithm 2 achieves a $\beta = 1/\left(1 - e^{-1/\alpha}\right) \approx \alpha + 1/2$ approximation to the optimal solution for the sum objective.

Algorithm 2: GreedySumOrienteering [2]

Input: Graph $G = (V, E)$, budget per tour B, number of required tours k, start/end node s
Output: Set of tours $\mathcal{P}$, visit times τ
$\mathcal{P} \leftarrow \emptyset$
$\tau \leftarrow \emptyset$
for $i = 1, 2, \ldots, k$ **do**
  $(P, \tau^P) \leftarrow$SingleTourOrienteering(G, B, s)
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{P\}$
  $\tau \leftarrow \tau \cup \{\tau^P\}$
  $G \leftarrow$RemoveNodesFromGraph$(G, V(P))$
return $(\mathcal{P}, \tau)$

Theorem 4.1. *[2] Given an α approximation algorithm for single-tour orienteering, Algorithm 2 achieves a $\beta = 1/\left(1 - e^{-1/\alpha}\right)$ approximation to $SUM(\mathcal{P}, \tau)$.*

Algorithm 2 does very well with respect to the sum objective, however, the output can be considered poor for a practical travel-planning setting. The reason is that itineraries tend to be front-loaded with the first few days visiting all important attractions, whereas the remaining days are poor in quality. A bad experience on one of the days can have a big impact on the perception of the quality of the entire trip, as a traveler may feel that a whole day of travel was wasted or unnecesarry to begin with. This motivates the study of max-min orienteering, which is the focus of the next section.

4.2 Max-Min Orienteering

In this section, we describe and analyze Algorithm 3, an approximation algorithm for max-min orienteering that loses a factor $\alpha + 2$, with α the approximation factor of single-tour orienteering. The algorithm receives as input the graph G (with costs for the edges and costs, utilities, and time windows for the nodes), the start/end node s, the per-tour budget B, the required number of tours k, and a target value T which all tours should achieve. If the algorithm manages to find k tours with at least value T, it returns them, otherwise it returns an empty solution. Let $\gamma = \alpha + 2$, with α the approximation factor of the single-tour case, and let OPT be the optimal value for $MIN(\mathcal{P}, \tau)$. We will prove that when $T = OPT/\gamma$, the algorithm is guaranteed to return k tours with value at least T, and, hence, achieves a γ approximation to the $MIN(\mathcal{P}, \tau)$ objective. We may run the algorithm multiple times with different guesses for T, using standard techniques (e.g., doubling the guess every time and later running binary search to pinpoint the best achievable value). The pseudocode is given in Algorithm 3. We now summarize the main stages. We will make the simplifying assumption that every node in the graph is reachable within distance $B/2$ from s (or equivalently that any node further than $B/2$ from s is removed from the graph).

Stage 1. In the first stage, the algorithm scans all nodes in the graph and removes every node that has utility at least T. Each one of these nodes, v, gives rise to a tour (s, v, s). Suppose the algorithm finds q such nodes. If $q \geq k$, we return this set of two-node tours. Otherwise we proceed to the following stages, with the assumption that $q < k$.

Algorithm 3: MaxMinOrienteering

Input: Graph $G = (V, E)$, budget per tour B, number of required tours k, start/end node s, target value T
Output: Set of tours $\mathcal{P}$, visit times τ
// Stage 1: Extract high utility nodes into tours
$\mathcal{P} \leftarrow \emptyset$
for $\forall v \in V : u_v \geq T$ **do**
  $\left(P, \tau^P\right) \leftarrow$MakeTwoNodeTour$(s, v)$
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{P\}$
  $\tau \leftarrow \tau \cup \{\tau^P\}$
  $G \leftarrow$RemoveNodeFromGraph(G, v)
$q \leftarrow |\mathcal{P}|$
if $q \geq k$ **then**
  return $\left(\{\mathcal{P}_1, \ldots, \mathcal{P}_k\}, \left\{\tau^{\mathcal{P}_1}, \ldots, \tau^{\mathcal{P}_k}\right\}\right)$
// Stage 2: Iteratively run single-tour and truncate
for $i = 1, 2, \ldots, k - q$ **do**
  $(P, \tau^P) \leftarrow$SingleTourOrienteering(G, B, s)
  $(P, \tau^P) \leftarrow$TruncateTour(P, τ^P, T)
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{P\}$
  $\tau \leftarrow \tau \cup \{\tau^P\}$
  $G \leftarrow$RemoveNodesFromGraph$(G, V(P))$
// Stage 3: Check the output and return
if $\forall P \in \mathcal{P}, U(P, \tau^P) \geq T$ **then**
  return $\left(\{\mathcal{P}_1, \ldots, \mathcal{P}_k\}, \left\{\tau^{\mathcal{P}_1}, \ldots, \tau^{\mathcal{P}_k}\right\}\right)$
else
  return $(\emptyset, \emptyset)$

Algorithm 4: TruncateTour

Input: Tour P, visit times τ^P, target value T
Output: Tour P, visit times τ^P
for $\forall v \in V(P)$ **do**
  $V(P) \leftarrow V(P) \setminus \{v\}$
  $\tau^P \leftarrow \tau^P \setminus \{\tau^P_v\}$
  if $\sum_{v \in V(P)} u_v \leq 2 \cdot T$ **then**
    break
return $\left(P, \tau^P\right)$

Stage 2. On the remainder of the graph, we run the single-tour algorithm $k - q$ times, and after each run we do the following: (i) keep removing a node from the tour (keeping the visit times of the remaining nodes unchanged) until the tour's value is $2 \cdot T$ or less, and (ii) update the graph by removing the nodes that are ultimately in the tour. Finally, before proceeding with the next iteration, we remove from the graph every node that was picked by our current iteration.

Stage 3. If Stage 2 outputs $k - q$ tours with value at least T each, return the union of tours output by Stages 1 and 2, otherwise return the empty set.

For the remainder of the section we prove the following result.

Theorem 4.2. *Given an α approximation for single-tour orienteering, running Algorithm 3 with $T = OPT/(\alpha + 2)$ achieves an $\alpha + 2$ approximation to $MIN(\mathcal{P}, \tau)$, with OPT the optimal max-min objective value.*

Proof. Let $\gamma = \alpha + 2$. If Stage 1 results in $q \geq k$, all tours have utility at least OPT/γ and, hence, the algorithm achieves a γ approximation. Now suppose $q < k$. We prove the following statement: Let $(\mathcal{P}^*, \tau^*)$ be the solution that optimizes $MIN(\mathcal{P}, \tau)$. At the end of iteration i of Stage 2 in Algorithm 3, the utility of $(\mathcal{P}^*, \tau^*)$ not picked by Algorithm 3 is at least

$$(k - q) \cdot OPT - \frac{2}{\gamma} \cdot i \cdot OPT \geq \left(1 - \frac{2}{\gamma}\right) \cdot (k - q) \cdot OPT. \quad (1)$$

We may see this as follow. After Stage 1 of Algorithm 3, we have removed q nodes. Even if we delete all tours from $(\mathcal{P}^*, \tau^*)$ that have any of these nodes, there will still be at least $k - q$ tours left and each one of them has value at least OPT, by definition. Hence, there is at least $(k - q) \cdot OPT$ utility left. Subsequently, each iteration of Stage 2, extracts value at most $2 \cdot OPT/\gamma$. Hence, the utility left is at least as in (1). We now prove that each of the tours returned by Stage 2 have value at least OPT/γ. By the fact that the utility in $(\mathcal{P}^*, \tau^*)$ not picked by the algorithm is always at least

$$\left(1 - \frac{2}{\gamma}\right) \cdot (k - q) \cdot OPT,$$

it follows that there is always one tour in $(\mathcal{P}^*, \tau^*)$ for which the algorithm has not picked

$$\left(1 - \frac{2}{\gamma}\right) \cdot OPT$$

of its utility. Since the single-tour orienteering algorithm that is used as a subroutine is an α approximation, it follows that the tour returned by single-tour orienteering will have value at least

$$\left(1 - \frac{2}{\gamma}\right) \cdot \frac{OPT}{\alpha} = \frac{OPT}{\gamma}.$$

To complete the proof, it remains to show that the process of truncating a tour will never reduce its utility to less than OPT/γ. Truncating a tour is performed by removing a node at a time, until the utility of the tour drops below $2 \cdot OPT/\gamma$. By the fact that, due to Stage 1, Stage 2 never sees nodes with utility more than OPT/γ, it follows that the last node removal of any tour will not take it to less than OPT/γ utility. □

Corollary 4.3. *For max-min orienteering on undirected graphs with no time windows, Algorithm 3 achieves a $4 + \epsilon$ approximation, for any $\epsilon > 0$. For all other models discussed at the start of the section, it achieves the same asymptotic guarantees as the single day case.*

5 Orienteering in Practice: Planning Itineraries

In this section we focus on the challenges of applying the orienteering model in a practical setting where we wish to plan tourist itineraries for a large number of cities in the world and on evaluating our method and decisions. One of the starting issues is that one needs to define a utility and cost model where the utility and duration of a node as well as other orienteering input parameters are extracted from historical visits and transit data. In Section 5.1, we explain our utility and cost model and how we apply it in two distinct datasets: a large dataset of Google data on 200 cities across the world and a smaller public set of Foursquare check-in data in New York City. We then proceed to experimentally evaluate the performance of our multi-tour algorithms for constructing itineraries in these real world cities in Section 5.2.

5.1 Data Extracted Input: Utility, Costs, and Time Windows

Google dataset. The main dataset we have at our disposal consists of a large number of anonymized historical visits to tourist attractions. The entries to the dataset include an arrival and a departure timestamp. Given this data, we wish to extract the utility of a place of interest (POI) and the typical duration of a visit to that POI. We adopt a clean and simple approach for both.

The utility of a place is expected to be an increasing function of the number of historical visits to the POI. Setting the utility to be equal to the number of historical visits is the simplest such function, but also has a natural interpretation; it is proportional to the probability that a random tourist visits the POI: Assuming no tourist visits the same POI more than once (we may fairly assume that this is a very rare occurrence), this probability is equal to the ratio of the POI's number of visits to the total number of tourists and hence proportional to the number of visits.

Understanding the typical amount of time tourists spend at a POI reduces to picking an aggregate that is a good representative of a set of given durations to the POI. We simply set the duration of a POI to be the median time spent at the place among all historical visits.

The travel times which are used as edge costs in our input to orienteering are given by a set of transit time predictions (provided by Google Maps) between all pairs of POIs and for various forms of transportation (e.g., walking, driving, public transit, any combination of those, etc.). We may fix a specific form of travel (e.g., walking only) or allow ourselves to select the smallest possible transit time between our designated endpoints. For the purposes of our experiments we allow any form of transit and use the quickest available option.

Finally, the node time windows are obtained by available data on the vast majority of POIs around the world (and which also surface on Google Maps). We set a start time for our tours (e.g. 9am) and the start and end of each time window are the corresponding offsets in minutes from the start time.

Foursquare data. We also evaluate our algorithms on a publicly available dataset of Foursquare check-ins [10, 27]. Entries in this dataset include location, category and timestamp. The utility model follows similarly to what we described above. Extracting durations of visits from this data is not possible, so we assign uniformly random durations in $[30, 60]$ minutes to the POIs.

Nature of instances. Given that transit time $t_{i,j}$ from POI i to POI j can differ from the transit time $t_{j,i}$ from POI j to POI i, and also given the presence of opening hours, real world cities manifest as instances of orienteering in directed graphs with time windows. However, the instances are far from adversarial. Transit times $t_{i,j}$ and $t_{j,i}$ can be different but in most cases their difference is small, while opening hours usually appear closer to deadline versions of orienteering (with most POIs open at the start of the tourist's day) or even orienteering without time windows (e.g., for

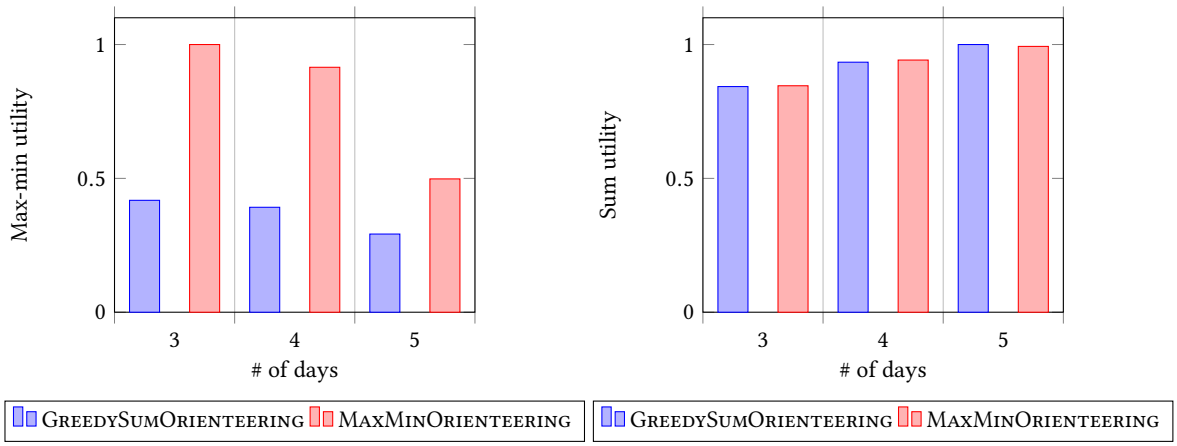

Figure 3: Max-min and sum comparison in New York City using Google data. Values have been normalized so that the largest is 1.

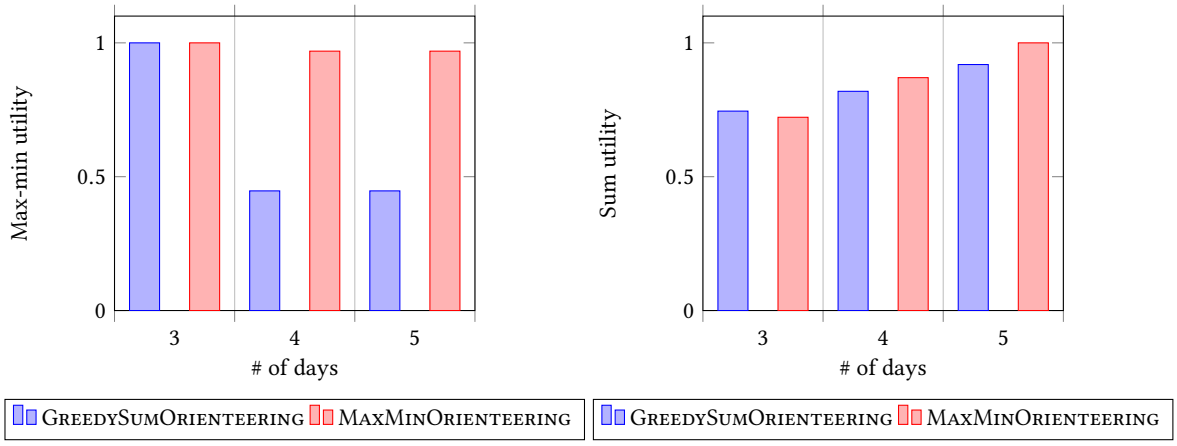

Figure 4: Max-min and sum comparison in New York City using Foursquare data. Values have been normalized so that the largest is 1.

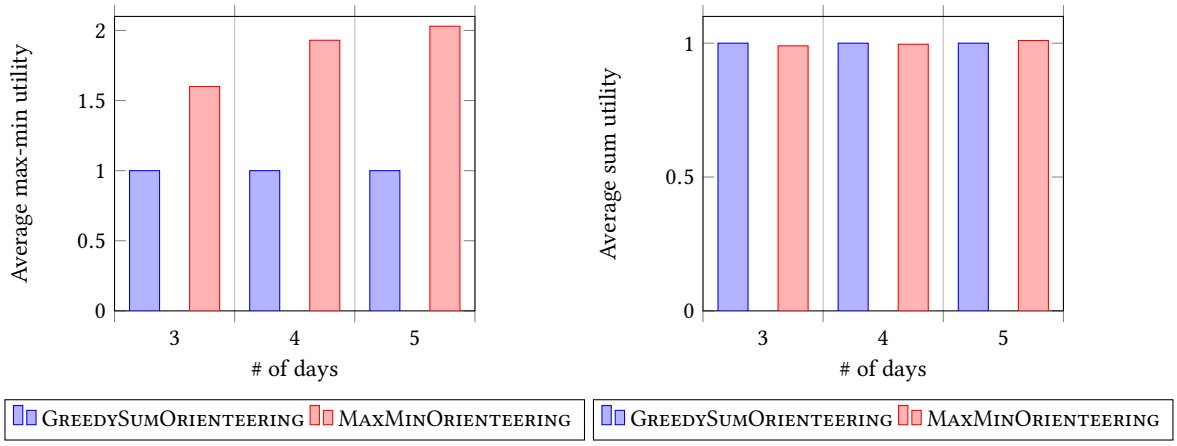

Figure 5: Average normalized max-min and sum utilities of the two algorithms across 200 cities using Google data. (By "normalized", we mean that the utilities in each city have been scaled so that GreedySumOrienteering achieves unit utility.)

10am-3pm itineraries, one would expect almost all POIs in the city to be open). In the evaluation that follows we will examine the performance of our multi-tour generating algorithms in hundreds of real world cities. As explained earlier, theoretical orienteering algorithms that are able to handle opening hours are very time consuming, especially with large graph sizes.

5.2 Evaluation

In this section we will experimentally evaluate our max-min orienteering multi-tour algorithm. Our first set of experiments shows that in real world cities under our utility and duration model, Algorithm 3 (MaxMinOrienteering) significantly improves on the performance of the literature-established multi-tour algorithm, Algorithm 2 (GreedySumOrienteering), with respect to the max-min objective, while incurring virtually no loss with respect to the sum objective. For our experiments, we generate itineraries that begin at 9am each day and have an 8 hour budget. For a starting location, we select some point near the center of the city. We note that, in our actual implementation, MaxMinOrienteering is supplemented by a post-processing stage where tours that still have enough budget left, may spend it and add more nodes via single-tour orienteering. Figures 3 and 4 present the relative performance of the two algorithms with respect to our two objectives in New York City using Google data and Foursquare [13] data respectively. Figure 5 aggregates the comparisons on Google data over 200 cities as follows: For each city, we normalize the utility values, both for max-min and for sum, so that GreedySumOrienteering achieves unit utility. The aggregate values for MaxMinOrienteering are then the averages across all cities.

We observe that our algorithm, MaxMinOrienteering, manages to improve the max-min objective to a large extent. For example, for the case with $k = 5$ days in the itinerary, the max-min solution is improved on average by a factor very close to 2, in comparison to GreedySumOrienteering, the standard multi-tour algorithm proposed in the literature. At the same time, there is virtually no loss with respect to the sum objective. In fact, the precise sum objective values for MaxMinOrienteering in Figure 5 are 0.99 for $k = 3$, 0.996 for $k = 4$, and 1.05 for $k = 5$. This suggests the performance of the two algorithms with respect to the sum objective is so close that, in some sets of instances, MaxMinOrienteering even beats GreedySumOrienteering.

Our second set of experiments consists of an evaluation of the produced itineraries by human raters. These evaluations put to the test both our algorithms and our model of extracting utilities and durations for the nodes. The rating methodology is as follows. Raters are randomly picked from a pool and are given the task of rating a multi-tour itinerary in a given city. Each city has either a two or three day itinerary. The rater has to score each separate day as *good* or *poor* and each city (i.e., multi-day itinerary) as *good*, *mediocre*, or *poor*. Each city is given to three raters. The score of each itinerary and each city is given by a majority rule. For cities, if all three scores are given, the itinerary is considered *mediocre*.

The human rater evaluation is presented in two parts. In the first part, we compare the ratings of our itineraries with two and three day itineraries that have been carefully curated by experts in 59 cities. The results exhibit that our performance is very close to that of travel experts and is presented in Figure 6. In the second part, we let raters evaluate our itineraries in a more challenging set of three day itineraries in 200 cities. The results are presented in Figure 7.

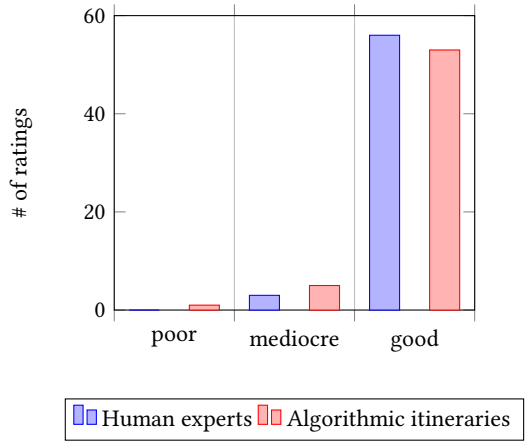

Figure 6: Evaluation by human raters for multi-day itineraries in 59 cities.

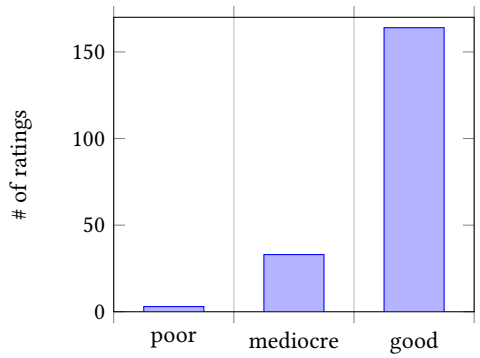

Figure 7: Evaluation of three-day algorithmic itineraries in 200 cities by human raters.

6 Conclusion

In our paper we describe an algorithm that aims to provide high quality multi-day tourist itineraries by maximizing the value of the worst day. Conceptually, the goal of this exercise is two-fold. First, do not let the travel experience be downgraded by the existence of a full day visiting low quality attractions. Second, try to impose an indirect limit on how much one can visit on a single day, thus keeping daily itineraries local and not revisiting the same neighborhoods across days. We complement the algorithm with an input model that uses real world data (Google transit time, hours, and historical visit data and public Foursquare check-ins) to construct orienteering instances. Our two sets of experiments served the purposes of, firstly, evaluating the performance of our max-min algorithm versus a standard baseline (numerical experiments) and, secondly, evaluating our overall approach of obtaining orienteering instances from data and solving them using our algorithm to produce multi-day travel itineraries (human rater evaluation).

Improving travel planning and the actual travel experience from a system perspective is still a fruitful area with many open challenges welcoming future work. On the algorithmic side, we conclude by proposing two interesting extensions. A first interesting direction is to extend our max-min algorithm to the case when the utility of a tour is a submodular function of the visited nodes. The sum algorithm is known to extend to this case but a similar extension of max-min is not straightforward. Another interesting problem is to extend both the sum and max-min algorithms to a setting where itineraries are generated for groups of people and the objective is to keep every one of them as happy as possible, by optimizing the minimum orienteering objective across all travelers.

References

[1] Nikhil Bansal, Avrim Blum, Shuchi Chawla, and Adam Meyerson. Approximation algorithms for deadline-TSP and vehicle routing with time-windows. In *Proceedings of STOC 2004.*

[2] Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. 2007. Approximation Algorithms for Orienteering and Discounted-Reward TSP. *SIAM J. Comput.* 37, 2 (2007), 653–670.

[3] Paolo Bolzoni, Sven Helmer, Kevin Wellenzohn, Johann Gamper, and Periklis Andritsos. Efficient itinerary planning with category constraints. In *Proceedings of SIGSPATIAL 2014.*

[4] Igo Brilhante, Jose Antonio Macedo, Franco Maria Nardini, Raffaele Perego, and Chiara Renso. Where shall we go today?: planning touristic tours with tripbuilder. In *Proceedings of CIKM 2013.*

[5] Igo Ramalho Brilhante, José Antônio Fernandes de Macêdo, Franco Maria Nardini, Raffaele Perego, and Chiara Renso. Where shall we go today? Planning touristic tours with TripBuilder. In *Proceedings of CIKM 2013.*

[6] Kamalika Chaudhuri, Brighten Godfrey, Satish Rao, and Kunal Talwar. Paths, Trees, and Minimum Latency Tours. In *Proceedings of FOCS 2003.*

[7] Chandra Chekuri, Nitish Korula, and Martin Pál. 2012. Improved algorithms for orienteering and related problems. *ACM Trans. Algorithms* 8, 3 (2012), 23:1–23:27.

[8] Chandra Chekuri and Martin Pál. A Recursive Greedy Algorithm for Walks in Directed Graphs. In *Proceedings of FOCS 2005.*

[9] Munmun De Choudhury, Moran Feldman, Sihem Amer-Yahia, Nadav Golbandi, Ronny Lempel, and Cong Yu. Automatic construction of travel itineraries using social breadcrumbs. In *Proceedings of Conference on Hypertext and Hypermedia 2010.*

[10] Géraud Le Falher, Aristides Gionis, and Michael Mathioudakis. Where Is the Soho of Rome? Measures and Algorithms for Finding Similar Neighborhoods in Cities. In *Proceedings of ICWSM 2015.*

[11] Shanshan Feng, Gao Cong, Bo, and Yeow Meng Chee. 2017. POI2Vec: Geographical Latent Representation for Predicting Future Visitors.

[12] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. Personalized Ranking Metric Embedding for Next New POI Recommendation.. In *IJCAI 2015.*

[13] Foursquare. 2017. Foursquare Database API. https://developer.foursquare.com/. (2017).

[14] Zachary Friggstad and Chaitanya Swamy. Compact, Provably-Good LPs for Orienteering and Regret-Bounded Vehicle Routing. In *Proceedings of IPCO 2017.*

[15] Naveen Garg. Saving an epsilon: a 2-approximation for the k-MST problem in graphs. In *Proceedings of STOC 2005.*

[16] Damianos Gavalas, Vlasios Kasapakis, Charalampos Konstantopoulos, Grammati Pantziou, Nikolaos Vathis, and Christos Zaroliagis. 2015. The eCOMPASS multimodal tourist tour planner. *Expert Systems with Applications* 42, 21 (2015), 7303–7316.

[17] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati E. Pantziou. 2014. A survey on algorithmic approaches for solving tourist trip design problems. *J. Heuristics* 20, 3 (2014), 291–328.

[18] Takeshi Kurashima, Tomoharu Iwata, Takahide Hoshide, Noriko Takaya, and Ko Fujimura. Geo topic model: joint modeling of user's activity area and interests for location recommendation. In *Proceedings of WSDM 2013.*

[19] Huayu Li, Yong Ge, and Hengshu Zhu. Point-of-Interest Recommendations: Learning Potential Check-ins from Friends. In *Proceedings of KDD 2016.*

[20] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. GeoMF: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of KDD 2014.*

[21] Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. 2015. Personalized Tour Recommendation Based on User Interests and Points of Interest Visit Durations.. In *Proceedings of IJCAI 2015.* 1778–1784.

[22] Bin Liu, Yanjie Fu, Zijun Yao, and Hui Xiong. Learning geographical preferences for point-of-interest recommendation. In *Proceedings of KDD 2013.*

[23] Yanchi Liu, Chuanren Liu, Bin Liu, Meng Qu, and Hui Xiong. Unified Point-of-Interest Recommendation with Temporal Interval Assessment. In *Proceedings of KDD 2016.*

[24] Claudio Lucchese, Raffaele Perego, Fabrizio Silvestri, Hossein Vahabi, and Rossano Venturini. How random walks can help tourism. In *Proceedings of ECIR 2012.*

[25] Cristina Ioana Muntean, Franco Maria Nardini, Fabrizio Silvestri, and Ranieri Baraglia. 2015. On learning prediction models for tourists paths. *ACM Transactions on Intelligent Systems and Technology (TIST)* 7, 1 (2015), 8.

[26] Daniele Quercia, Rossano Schifanella, and Luca Maria Aiello. The shortest path to happiness: recommending beautiful, quiet, and happy routes in the city. In *Proceedings of Conference on Hypertext and Social Media 2014.*

[27] Vineeth Rakesh, Niranjan Jadhav, Alexander Kotov, and Chandan K Reddy. Probabilistic Social Sequential Model for Tour Recommendation. In *Proceedings of WSDM 2017.*

[28] Senjuti Basu Roy, Gautam Das, Sihem Amer-Yahia, and Cong Yu. Interactive itinerary planning. In *Proceedings of ICDE 2011.*

[29] Wouter Souffriau and Pieter Vansteenwegen. Tourist trip planning functionalities: State–of–the–art and future. In *Proceedings of International Conference on Web Engineering, 2010.*

[30] Yeran Sun, Hongchao Fan, Mohamed Bakillah, and Alexander Zipf. 2015. Road-based travel recommendation using geo-tagged images. *Computers, Environment and Urban Systems* 53 (2015), 110–122.

[31] Pradeep Varakantham, Hala Mostafa, Na Fu, and Hoong Chuin Lau. DIRECT: A Scalable Approach for Route Guidance in Selfish Orienteering Problems. In *Proceedings of International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2015.*

[32] Xiaoting Wang, Christopher Leckie, Jeffrey Chan, Kwan Hui Lim, and Tharshan Vaithianathan. Improving Personalized Trip Recommendation by Avoiding Crowds. In *Proceedings of CIKM 2016.*

[33] Zhijun Yin, Liangliang Cao, Jiawei Han, Jiebo Luo, and Thomas S. Huang. Diversified Trajectory Pattern Ranking in Geo-tagged Social Media. In *Proceedings of SDM 2011.*

[34] Hyoseok Yoon, Yu Zheng, Xing Xie, and Woontack Woo. 2012. Social itinerary recommendation from user-generated digital trails. *Personal and Ubiquitous Computing* 16, 5 (2012), 469–484.

[35] Yan-Tao Zheng, Zheng-Jun Zha, and Tat-Seng Chua. 2012. Mining Travel Patterns from Geotagged Photos. *ACM TIST* 3, 3 (2012), 56:1–56:18.